

讲堂 > Linux性能优化实战 > 文章详情

09 | 基础篇：怎么理解Linux软中断？

2018-12-10 倪朋飞



09 | 基础篇：怎么理解Linux软中断？

朗读人：冯永吉 09'35" | 8.79M

你好，我是倪朋飞。

上一期，我用一个不可中断进程的案例，带你学习了 `iowait`（也就是等待 I/O 的 CPU 使用率）升高时的分析方法。这里你要记住，进程的不可中断状态是系统的一种保护机制，可以保证硬件的交互过程不被意外打断。所以，短时间的不可中断状态是很正常的。

但是，当进程长时间都处于不可中断状态时，你就得当心了。这时，你可以使用 `dstat`、`pidstat` 等工具，确认是不是磁盘 I/O 的问题，进而排查相关的进程和磁盘设备。关于磁盘 I/O 的性能问题，你暂且不用专门去背，我会在后续的 I/O 部分详细介绍，到时候理解了也就记住了。

其实除了 `iowait`，软中断（`softirq`）CPU 使用率升高也是最常见的一种性能问题。接下来的两节课，我们就来学习软中断的内容，我还会以最常见的反向代理服务器 Nginx 的案例，带你分析这种情况。

从“取外卖”看中断

说到中断，我在前面[关于“上下文切换”的文章](#)，简单说过中断的含义，先来回顾一下。中断是系统用来响应硬件设备请求的一种机制，它会打断进程的正常调度和执行，然后调用内核中的中断处理程序来响应设备的请求。

你可能要问了，为什么要有中断呢？我可以举个生活中的例子，让你感受一下中断的魅力。

比如说你订了一份外卖，但是不确定外卖什么时候送到，也没有别的方法了解外卖的进度，但是，配送员送外卖是不等人的，到了你这儿没人取的话，就直接走人了。所以你只能苦苦等着，时不时去门口看看外卖送到没，而不能干其他事情。

不过呢，如果在订外卖的时候，你就跟配送员约定好，让他送到后给你打个电话，那你就不用苦苦等待了，就可以去忙别的事情，直到电话一响，接电话、取外卖就可以了。

这里的“打电话”，其实就是一个中断。没接到电话的时候，你可以做其他的事情；只有接到了电话（也就是发生中断），你才要进行另一个动作：取外卖。

这个例子你就可以发现，**中断其实是一种异步的事件处理机制，可以提高系统的并发处理能力。**

由于中断处理程序会打断其他进程的运行，所以，**为了减少对正常进程运行调度的影响，中断处理程序就需要尽可能快地运行。**如果中断本身要做的事情不多，那么处理起来也不会有太大问题；但如果中断要处理的事情很多，中断服务程序就有可能要运行很长时间。

特别是，中断处理程序在响应中断时，还会临时关闭中断。这就会导致上一次中断处理完成之前，其他中断都不能响应，也就是说中断有可能会丢失。

那么还是以取外卖为例。假如你订了 2 份外卖，一份主食和一份饮料，并且是由 2 个不同的配送员来配送。这次你不用时时等待着，两份外卖都约定了电话取外卖的方式。但是，问题又来了。

当第一份外卖送到时，配送员给你打了个长长的电话，商量发票的处理方式。与此同时，第二个配送员也到了，也想给你打电话。

但是很明显，因为电话占线（也就是关闭了中断响应），第二个配送员的电话是打不通的。所以，第二个配送员很可能试几次后就走掉了（也就是丢失了一次中断）。

软中断

如果你弄清楚了“取外卖”的模式，那对系统的中断机制就很容易理解了。事实上，为了解决中断处理程序执行过长和中断丢失的问题，Linux 将中断处理过程分成了两个阶段，也就是**上半部和下半部**：

- **上半部用来快速处理中断**，它在中断禁止模式下运行，主要处理跟硬件紧密相关的或时间敏感的工作。

- **下半部用来延迟处理上半部未完成的工作，通常以内核线程的方式运行。**

比如说前面取外卖的例子，上半部就是你接听电话，告诉配送员你已经知道了，其他事儿见面再说，然后电话就可以挂断了；下半部才是取外卖的动作，以及见面后商量发票处理的动作。

这样，第一个配送员不会占用你太多时间，当第二个配送员过来时，照样能正常打通你的电话。

除了取外卖，我再举个最常见的网卡接收数据包的例子，让你更好地理解。

网卡接收到数据包后，会通过**硬件中断**的方式，通知内核有新的数据到了。这时，内核就应该调用中断处理程序来响应它。你可以自己先想一下，这种情况下的上半部和下半部分别负责什么工作呢？

对上半部来说，既然是快速处理，其实就是要把网卡的数据读到内存中，然后更新一下硬件寄存器的状态（表示数据已经读好了），最后再发送一个**软中断**信号，通知下半部做进一步的处理。

而下半部被软中断信号唤醒后，需要从内存中找到网络数据，再按照网络协议栈，对数据进行逐层解析和处理，直到把它送给应用程序。

所以，这两个阶段你也可以这样理解：

- 上半部直接处理硬件请求，也就是我们常说的硬中断，特点是快速执行；
- 而下半部则是由内核触发，也就是我们常说的软中断，特点是延迟执行。

实际上，上半部会打断 CPU 正在执行的任务，然后立即执行中断处理程序。而下半部以内核线程的方式执行，并且每个 CPU 都对应一个软中断内核线程，名字为 “ksoftirqd/CPU 编号”，比如说，0 号 CPU 对应的软中断内核线程的名字就是 ksoftirqd/0。

不过要注意的是，软中断不只包括了刚刚所讲的硬件设备中断处理程序的下半部，一些内核自定义的事件也属于软中断，比如内核调度和 RCU 锁（Read-Copy Update 的缩写，RCU 是 Linux 内核中最常用的锁之一）等。

那要怎么知道你的系统里有哪些软中断呢？

查看软中断和内核线程

不知道你还记不记得，前面提到过的 proc 文件系统。它是一种内核空间和用户空间进行通信的机制，可以用来查看内核的数据结构，或者用来动态修改内核的配置。其中：

- /proc/softirqs 提供了软中断的运行情况；
- /proc/interrupts 提供了硬中断的运行情况。

运行下面的命令，查看 `/proc/softirqs` 文件的内容，你就可以看到各种类型软中断在不同 CPU 上的累积运行次数：

```
1 $ cat /proc/softirqs
2
3          CPU0      CPU1
4      HI:          0          0
5      TIMER:      811613    1972736
6      NET_TX:       49         7
7      NET_RX:    1136736    1506885
8      BLOCK:        0         0
9      IRQ_POLL:     0         0
10     TASKLET:    304787     3691
11     SCHED:     689718    1897539
12     HRTIMER:     0         0
13     RCU:      1330771    1354737
```

[复制代码](#)

在查看 `/proc/softirqs` 文件内容时，你要特别注意以下这两点。

第一，要注意软中断的类型，也就是这个界面中第一列的内容。从第一列你可以看到，软中断包括了 10 个类别，分别对应不同的工作类型。比如 `NET_RX` 表示网络接收中断，而 `NET_TX` 表示网络发送中断。

第二，要注意同一种软中断在不同 CPU 上的分布情况，也就是同一行的内容。正常情况下，同一种中断在不同 CPU 上的累积次数应该差不多。比如这个界面中，`NET_RX` 在 CPU0 和 CPU1 上的中断次数基本是同一个数量级，相差不大。

不过你可能发现，`TASKLET` 在不同 CPU 上的分布并不均匀。`TASKLET` 是最常用的软中断实现机制，每个 `TASKLET` 只运行一次就会结束，并且只在调用它的函数所在的 CPU 上运行。

因此，使用 `TASKLET` 特别简便，当然也会存在一些问题，比如说由于只在一个 CPU 上运行导致的调度不均衡，再比如因为不能在多个 CPU 上并行运行带来了性能限制。

另外，刚刚提到过，软中断实际上是以内核线程的方式运行的，每个 CPU 都对应一个软中断内核线程，这个软中断内核线程就叫做 `ksoftirqd/CPU 编号`。那要怎么查看这些线程的运行状况呢？

其实用 `ps` 命令就可以做到，比如执行下面的指令：

```
1 $ ps aux | grep softirq
2 root      7  0.0  0.0   0   0 ?        S   Oct10   0:01 [ksoftirqd/0]
3 root     16  0.0  0.0   0   0 ?        S   Oct10   0:01 [ksoftirqd/1]
```

[复制代码](#)

注意，这些线程的名字外面都有中括号，这说明 ps 无法获取它们的命令行参数（cmline）。一般来说，ps 的输出中，名字括在中括号里的，一般都是内核线程。

小结

Linux 中的中断处理程序分为上半部和下半部：

- 上半部对应硬件中断，用来快速处理中断。
- 下半部对应软中断，用来异步处理上半部未完成的工作。

Linux 中的软中断包括网络收发、定时、调度、RCU 锁等各种类型，可以通过查看 /proc/softirqs 来观察软中断的运行情况。

思考

最后，我想请你一起聊聊，你是怎么理解软中断的？你有没有碰到过因为软中断出现的性能问题？你又是怎么分析它们的瓶颈的呢？你可以结合今天的内容，总结自己的思路，写下自己的问题。

欢迎在留言区和我讨论，也欢迎把这篇文章分享给你的同事、朋友。我们一起在实战中演练，在交流中进步。



©版权归极客邦科技所有，未经许可不得转载

上一篇 08 | 案例篇：系统中出现大量不可中断进程和僵尸进程怎么办？（下）

下一篇 10 | 案例篇：系统的软中断CPU使用率升高，我该怎么办？

精选留言



我来也
[D9打卡]

👍 15

=====

问题:怎么理解软中断?

我的理解比较简单粗暴,硬中断是硬件产生的,比如键盘、鼠标的输入,硬盘的写入读取、网卡有数据了;软中断是软件产生的,比如程序内的定时器、[文中提到的RCU锁]。

再加上今天的上半部下半部,更好的理解了网卡的处理实际是有硬中断和软中断的。

=====

问题:有没有碰到过因为软中断出现的性能问题?

有,且是血淋淋的教训。

之前的c程序用到了别人写的动态库[如:lib.a],在物理机上,进程的cpu利用率在0%;而切换到了云服务器,即使空载,单进程的cpu利用率都有30%+。

以前没经验嘛,各种自查,无结果,但是自己的进程cpu利用率又那么高,总是不安心。

后来才通过vmstat 检测到系统的软中断每秒有100W+次。

最后各自百度,找人,才发现是那个动态库在处理网络收发消息时,使用了usleep(1)来休息,每次休息1纳秒,单次中断的耗时都不止1纳秒。

如果是现在,我会如下分析:

- 1.检测是哪个线程占用了cpu: top -H -p XX 1 / pidstat -wut -p XX 1
- 2.在进程中打印各线程号. 找到是哪个线程.[此过程也可以省略 但可以快速定位线程]
- 3.第一步应该可以判断出来中断数过高. 再使用 cat /proc/softirqs 查看是哪种类型的中断数过高.
- 4.不知道perf report -g -p XX 是否可以定位到具体的系统调用函数.
- 5.最终还是要查看源码,定位具体的位置,并加以验证.

感觉现在随便怎么分析都可以快速定位到是动态库的锅。

想当初可是好几个月都无能为为力啊,最后还是几个人各种查,搞了一周多才定位到原因。

最后再吐槽下,没有root权限的普通账户真是不方便啊,有些工具只能安装在自己的目录下,还有些好用的工具根本无权限运行,哎...

2018-12-10

作者回复

嗯嗯, 已经是很有经验的老手了 😊

大多数情况下 root 权限都是必须的, 还是准备个root权限的环境实践吧

2018-12-10



Linuxer

👍 6

经常听同事说大量的网络小包会导致性能问题, 一直不太理解, 从今天的课程来看, 是不是大量的小网络包会导致频繁的硬中断和软中断呢? 希望老师给予指点, 谢谢

2018-12-10

作者回复

正解

2018-12-10



每天晒白牙

【D9打卡】

主题:软中断

中断:系统用来响应硬件设备请求的一种机制, 会打断进程的正常调度和执行, 通过调用内核中的中断处理程序来响应设备的请求。

1.中断是一种异步的事件处理机制, 能提高系统的并发处理能力

2.为了减少对正常进程运行进行影响, 中断处理程序需要尽快运行。

3.中断分为上下两个部分

(1)上部分用来快速处理中断, 在中断禁止模式下, 主要处理跟硬件紧密相关的或时间敏感的工作

(2)下部分用来延迟处理上半部分未完成的工作, 通常以内核线程的方式运行。

小结:

上半部分直接处理硬件请求, 即硬中断, 特点是快速执行

下部分由内核触发, 即软中断, 特点是延迟执行

软中断除了上面的下部分, 还包括一些内核自定义的事件, 如:内核调度 RCU锁 网络收发 定时等

软中断内核线程的名字:ksoftirq/cpu编号

4.proc文件系统是一种内核空间和用户空间进行通信的机制, 可以同时用来查看内核的数据结构又能用了动态修改内核的配置, 如:

/proc/softirqs 提供软中断的运行情况

/proc/interrupts 提供硬中断的运行情况

2018-12-10

作者回复



2018-12-10



Eric

中断不是可以嵌套的吗? 在中断处理程序中可以开中断以响应更高优先级的中断, 为什么第二次中断会丢失? 是指中断隐指令过程吗? ??

2018-12-10



ninuxer

打卡, day10

用外卖的例子, 延伸到网卡的例子, 非常形象, ㊗

2018-12-10

作者回复



2018-12-10

4

4

3



Good Good Study

👍 2

期待第二篇，我有个业务48核心，怎么调整都是只用了前面24核心的软中断，期待更新

2018-12-10



胡萝卜
打卡

👍 1

2018-12-10



Days

👍 1

而下半部以内核线程的方式执行，并且每个 CPU 都对应一个软中断内核线程，这里我觉得不是所有软中断直接被ksoftirqd处理，只有大量软中断产生，或者处理软终端超时才唤醒ksoftirqd线程

2018-12-10



solar

👍 1

打卡，又学到一些很细节的东西，感动

2018-12-10



湖湘志
D9

👍 1

2018-12-10



张强

👍 0

倪老师，您好。我今天遇到个问题，服务端监听程序收到的客户端的连接状态使用netstat命令看到有几个syn_recv状态（少量），我使用telnet新的连接来测试也是这个状态。客户端的请求数量规模很小，在个位数。这种问题的解决思路是什么？很难出现，重启后好了，其他监听端口也没问题。

2018-12-12



春暖花开

👍 0

是否可以将中断处理专门属于一个核心cpu处理，通过亲和性

2018-12-12

作者回复

可以的

2018-12-13



小老鼠

👍 0

软中断时间太长会不会影响性能

2018-12-12

作者回复

会的

2018-12-13



dexter

0

打卡

2018-12-12

作者回复



2018-12-13



kakasir

0

用外卖的例子形象的说明了中断，很清晰，我可以比一个不大恰当的例子，就是同步和异步，是不是就像硬中断和软中断

2018-12-12

作者回复

嗯嗯 不太准确。两种中断的触发都是异步的，只是处理的任务多少问题

2018-12-13



魏峰

0

scp小文件很慢的原因是网络小包的软中断？

2018-12-11

作者回复

除了软中断，更大的问题是文件IO

2018-12-13



Wendy

0

答复wykkx，兄弟，你用pidstat -t 1 查看一下java线程嘛，你说的有点类似于短时进程，ps看看进程状态有哪些，是否有僵尸进程？是否java程序在大量的调用短时函数(进程)？。我们目前学了以下几个指标:cpu使用率、iowait、上下文切换、软硬中断。你比照一下这几个指标。

2018-12-11

作者回复

谢谢分享和讨论

2018-12-13



风萧雨瑟

0

老师我这里有一个问题请教一下：mysql的一台新机器:CentOS 7.5，32c 64G内存。ssd盘。将mysql 参数innodb_flush_log_at_trx_commit设置为1的时候会出现mysql复制延迟增大，iostat观察%util能达到50%。而同样的其它从库，机器规格,mysql参数都一样。但却不会出这样的情况。iostat只有百分之几。但将innodb_flush_log_at_trx_commit调整为2的时候复制就完全正常了，%util也降到正常水平。innodb_flush_log_at_trx_commit是会影响刷盘操作。但其它机器都没有出现这种情况。对ssd硬盘我通过fio做测试，iops随机读写结果跟其它机器上的ssd盘区别不大。并且，这台机器上的两块ssd我都测试过。情况一样。不正常的这台机器iostat结果

```
iostat -d sda1 -x 2 -m
```

```
Linux 3.10.0-862.9.1.el7.x86_64 (g1-db-rw-15) 12/11/2018 _x86_64_ (32 CPU)
```

```
Device r/s w/s rMB/s wMB/s rrqm/s wrqm/s %rrqm %wrqm r_await w_await aqu-sz r
areq-sz wareq-sz svctm %util
sda1 7.50 268.00 0.12 2.25 0.00 0.00 0.00 0.00 1.00 0.84 0.23 16.00 8.61 0.77 21.35
```

```
Device r/s w/s rMB/s wMB/s rrqm/s wrqm/s %rrqm %wrqm r_await w_await aqu-sz r
areq-sz wareq-sz svctm %util
sda1 4.98 387.56 0.08 2.98 0.00 0.00 0.00 0.00 0.40 0.93 0.36 16.00 7.87 0.79 31.09
```

正常的从库iostat结果：

```
iostat -d sda1 -x 2 -m
```

```
Linux 2.6.32-431.el6.x86_64 (g1-db-cat-02) 12/11/2018 _x86_64_ (32 CPU)
```

```
Device: rrqm/s wrqm/s r/s w/s rMB/s wMB/s avgrq-sz avgqu-sz await svctm %util
sda1 0.00 0.00 13.50 339.00 0.21 4.06 24.84 0.07 0.19 0.11 3.95
```

```
Device: rrqm/s wrqm/s r/s w/s rMB/s wMB/s avgrq-sz avgqu-sz await svctm %util
sda1 0.00 0.00 11.00 268.00 0.17 3.81 29.20 0.05 0.18 0.14 4.00
```

这两个机器因为系统不一样。iostat内容显示略有不同，有问题的这台我升级过sysstat，升级之前iostat显示内容基本一样。

有一点很疑惑：两台规格一样的机器。都是从库，都是独点一块ssd，ssd fio测试也无多大区别。为何写入性能上差别很大。

通过perf来抓取看的话。有问题的这台显示：

```
libpthread-2.17.so [.] start_thread - start_thread - 63.21% io_handler_thread - 63.21%
fil_aio_wait - 45.23% os_aio_handler - 28.35% os_file_write_page 因为这台从库上没有负载，单纯只是个从库。
```

```
top
```

```
%Cpu19 : 3.4 us, 2.0 sy, 0.0 ni, 58.4 id, 36.2 wa, 0.0 hi, 0.0 si, 0.0 st
```

```
%CPU: 17.2
```

```
2018-12-11
```



wykkx

0

老师我们线上系统最近有个问题，4核的cpu，load5和load15总是过一段时间就超过4*0.8的值，这个时候我用top看系统的情况，没有D进程，也没有一直R的进程，cpu利用率很低不到5%，cs的值也就几千，iostat查看util和iowait的值都不高，vmstat中r和b这两列基本都是0，偶尔r会有1。这台机器上就一个java进程，请教下次复现该问题时，我该如何排查？谢谢

```
2018-12-11
```

作者回复

最好是把历史的运行情况监控起来，比如部署监控系统，或者最简单的把这些命令放到脚本里定期执行，这样才有依据分析历史

```
2018-12-11
```



西果果树

0

```
# cat /proc/softirqs
```

```
CPU0
```

```
HI: 1
```

```
TIMER: 52670170
```

NET_TX: 9

NET_RX: 4737439

BLOCK: 557149

BLOCK_IOPOLL: 0

TASKLET: 16

SCHED: 0

HRTIMER: 0

RCU: 32239865

这样的情况，是存在问题吗？

2018-12-11

作者回复

没问题，这是只有一个CPU

2018-12-11